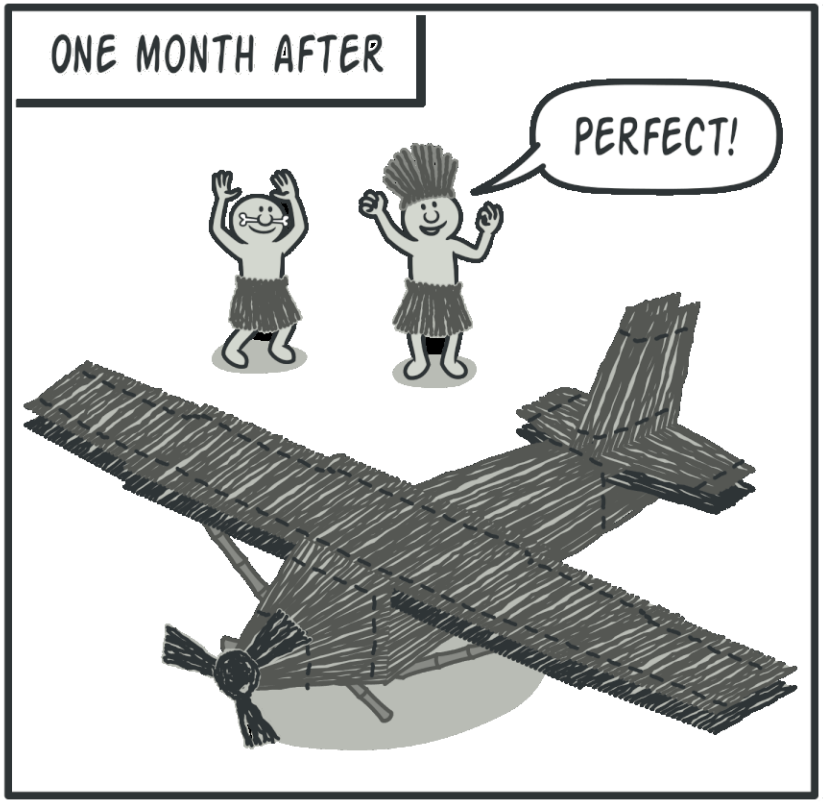
Prototype

Also known as: Clone

Prototype is a creational design pattern that lets you copy existing objects without making your code dependent on their classes

Problem

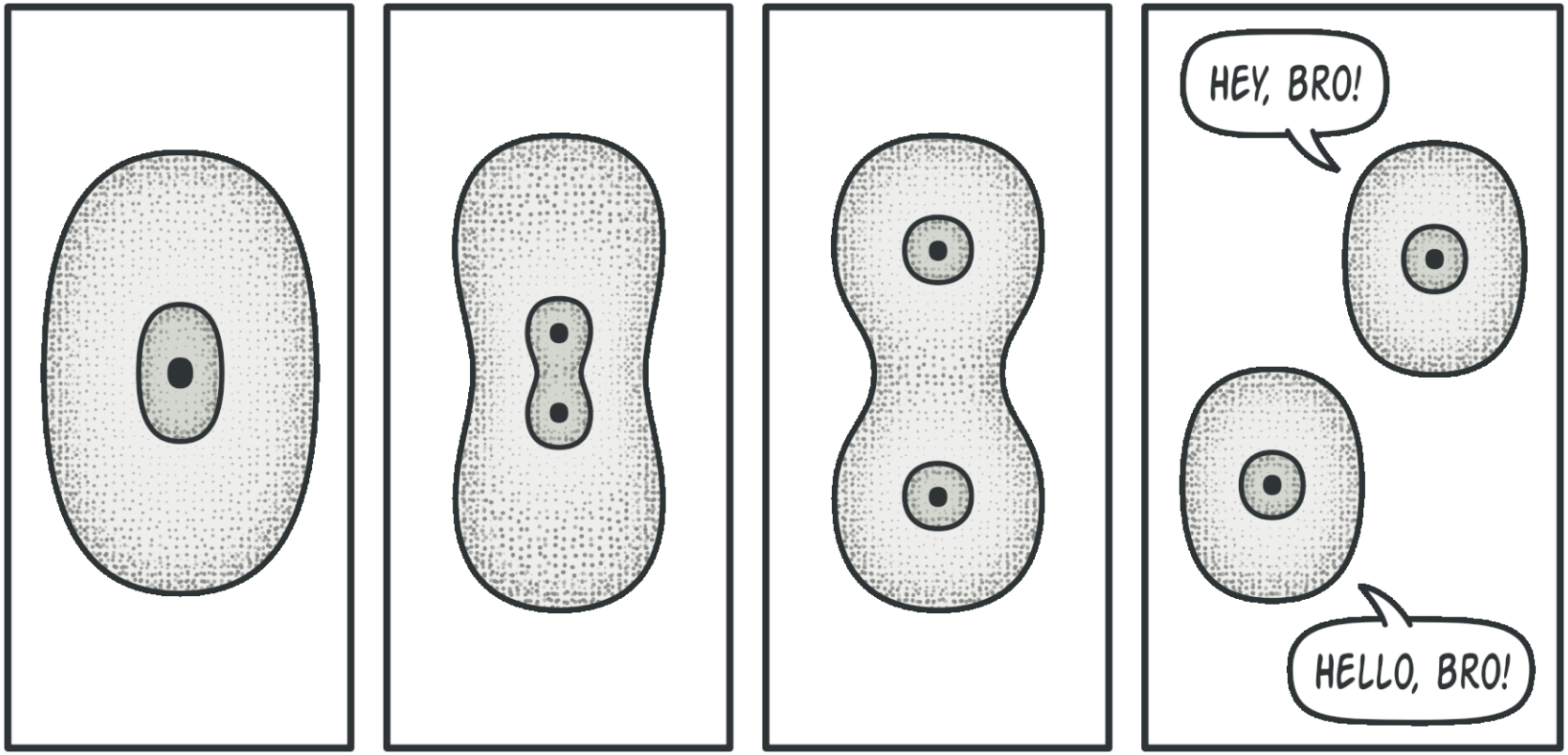


Copying an object “from the outside” **isn’t** always possible

Solution

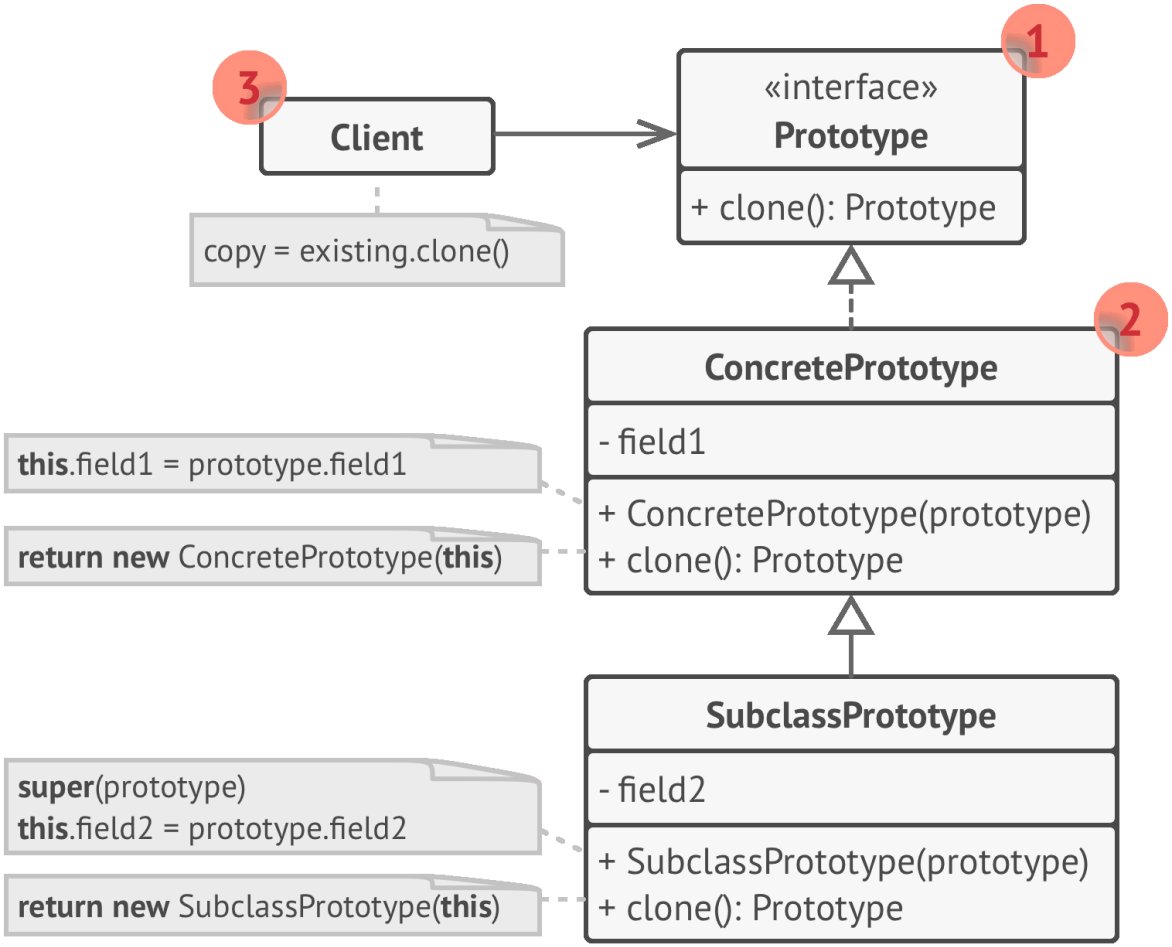
Pre-built prototypes can be an alternative to sub classing

Example

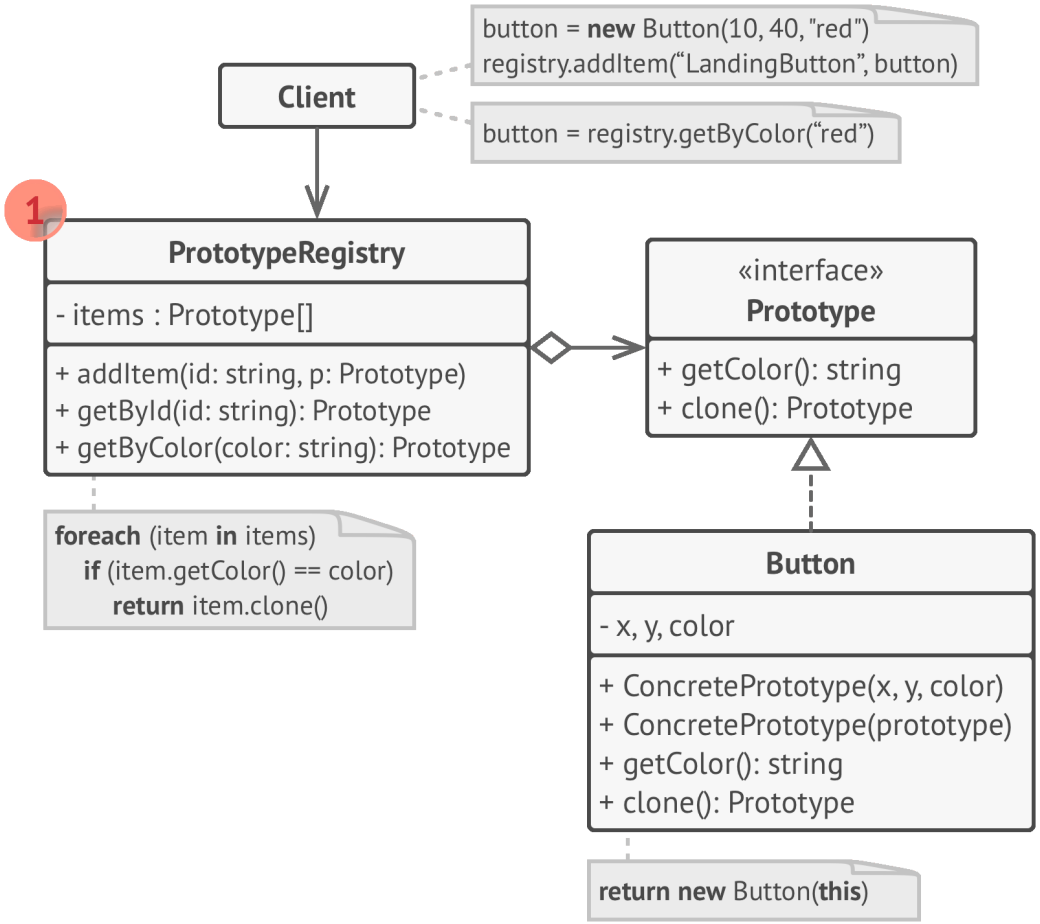


The division of a cell.

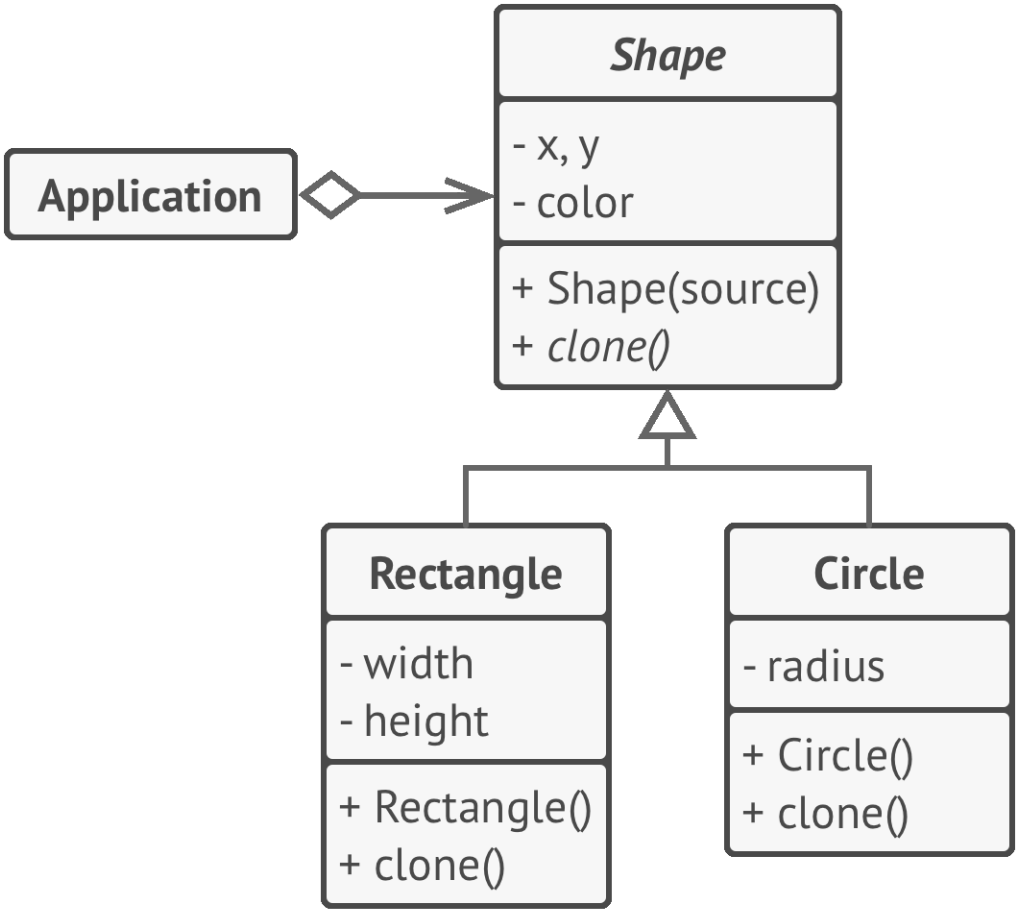
Structure


1
«interface»
Prototype
+ clone(): Prototype
3
Client
copy = existing.clone()
2
ConcretePrototype
- field1
+ ConcretePrototype(prototype)
+ clone(): Prototype
this.field1 = prototype.field1
return new ConcretePrototype(this)
SubclassPrototype
- field2
+ SubclassPrototype(prototype)
+ clone(): Prototype
super(prototype)
this.field2 = prototype.field2
return new SubclassPrototype(this)

Prototype registry implementation



Implementation



Cloning a set of objects that belong to a class hierarchy.

Applicability

- The code shouldn’t depend on the concrete classes of objects that is needed to copy.

- To reduce the number of subclasses that only differ in the way they initialize their respective objects.

Pros and Cons

- We can clone objects without coupling to their concrete classes.
- We can get rid of repeated initialization code in favor of cloning pre-built prototypes.
- We can produce complex objects more conveniently.
- We get an alternative to inheritance when dealing with configuration presets for complex objects.
- Cloning complex objects that have circular references might be very tricky.